CMPS 312

Read Chapters
3, 6, 14, 22

UI Components & Layout

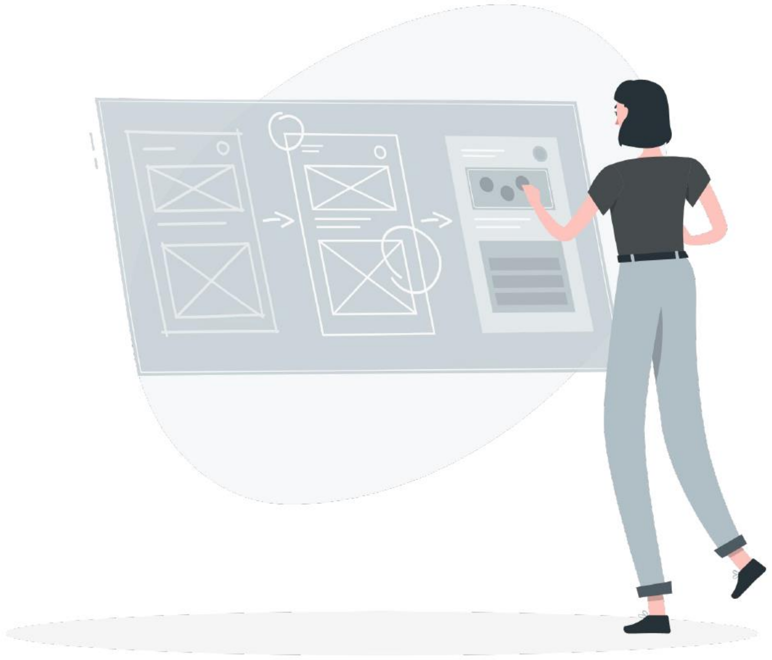
Dr. Abdelkarim Erradi

CSE@QU

Outline

1. Activity
2. UI Components
3. Constraint Layout

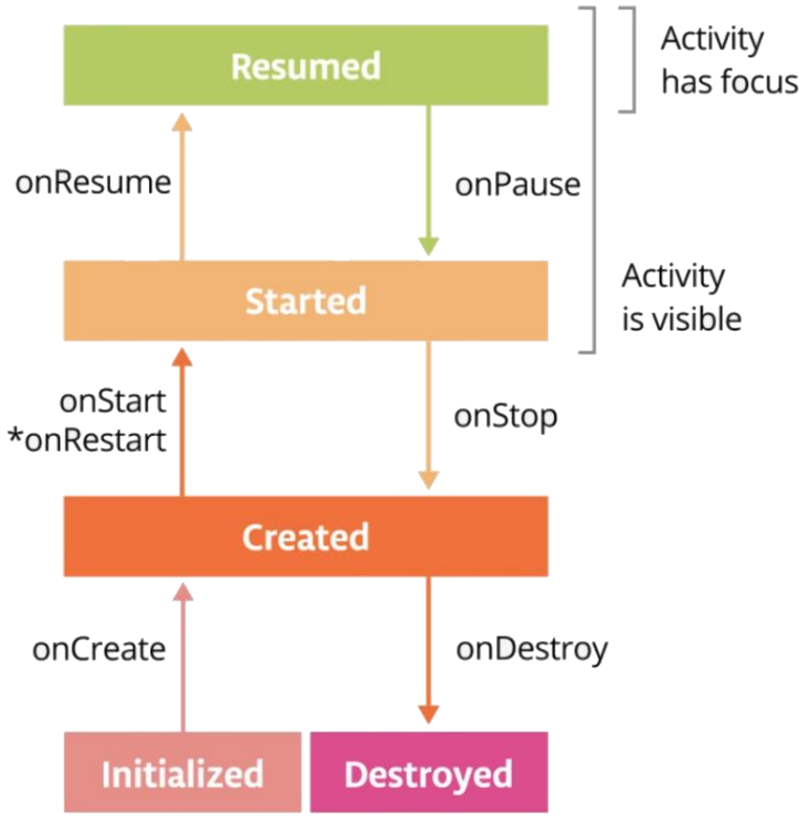
Activity



Activity

- **Activity** provides the UI that the user interacts with
  - Allow the user to do something such as order groceries, send email
  - Has layout (.xml) file & Activity class (aka **UI Controller**)
  - This allows a **clear separation** between the UI and the app logic
- Connecting the activity UI Controller with the layout is done in the **onCreate** method
- Activity class (**UI Controller**) defines listeners to handle events:
  - User interaction events such press a button or enters text in an EditText
  - External events such as screen rotation or receiving a notification

Activity Lifecycle



An activity has essentially **four states**:

- ***Resumed*** if the activity in the foreground of the screen (has focus)
- ***Started*** if the activity has lost focus but is still visible (e.g., beneath a dialog box).
  - When the user returns to the activity, it is **resumed**
- **Created** if the activity is completely obscured by another activity.
  - When the user navigates to the activity, it must be **restarted** and restored to its previous state.
- **Destroyed** when the user closes the app or if the activity is killed (when memory is needed or due to finish() being called on the activity)

Activity Lifecycle

onCreate()
Created but not yet active on the screen

→

onStart()
Activity is visible but not quite ready to receive focus or interaction

→

onResume()
Activity is visible and active in the app foreground

onPause()
Activity is visible, but something has taken foreground priority

→

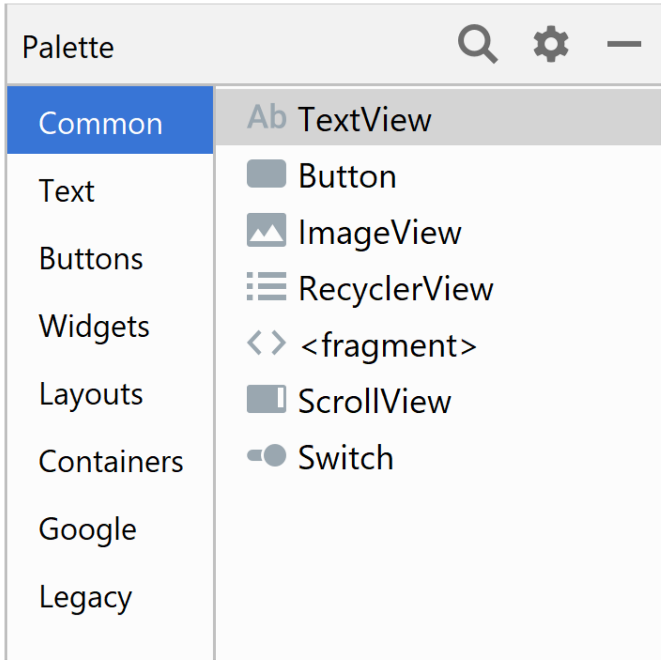
onStop()
Activity is no longer visible on screen

→

onDestroy()
Activity is about to be destroyed because user navigated away and OS needs resources

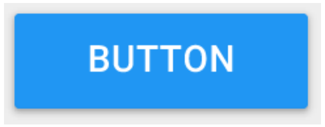
- Events handlers can be associated to these events
  - Android invokes them when the activity moves from one state to another
  - E.g., in onCreate() you inflate the layout and define click listeners

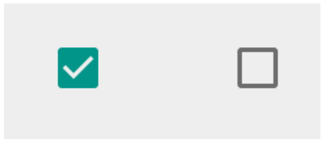
What Makes up an Activity UI?

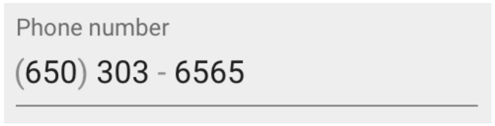


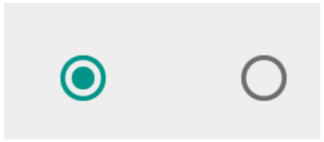
- **Views**
  - Set of pre-built UI components that can be composed to create a GUI
  - e.g. Button, TextView, Menu, List, etc.
- **Layout**
  - Container that controls the size and positioning of views in the Activity

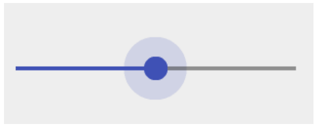
UI Components

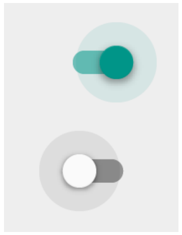
Button


CheckBox


EditText


RadioButton


SeekBar


Switch


< Back

Views

- **View = Widget = Control**
  - Examples: Button, Switch, Spinner, TextView, EditText, ImageView
  - Advanced Views (covered later): **RecyclerView** & MapView

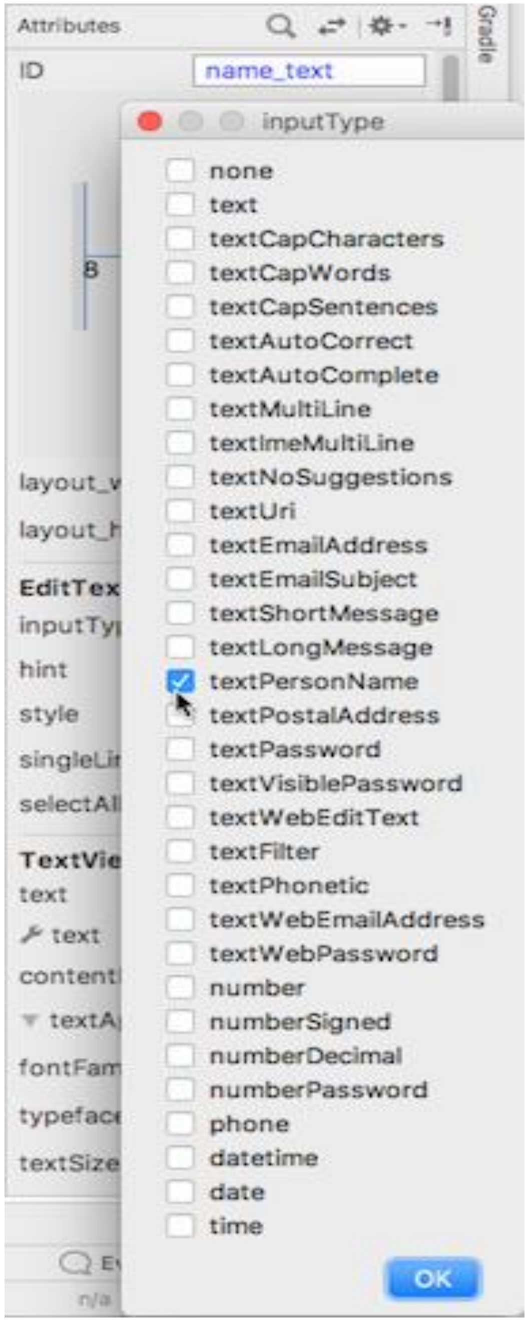
- **Common Attributes**
  - id (i.e. **android:id="@+id/myViewId"**)
  - `layout_width`, `layout_height`
    - Values: `match_constraint` (or 0dp), `wrap_content`, fixed size (e.g., 50dp)

Views (Attributes and Listeners)

- TextView - Displays text on the screen
  - `text`
- EditText - Allows entering user input
  - `inputType   :` such as email, phone number, etc.
  - `text`
  - `.addTextChangedListener { … }`
- Button - Clickable view responding to user clicks
  - `text`
  - `.setOnClickListener { … }`
- ImageView - Displays image from a URL or from a resource file
  - `.setImageDrawable(drawable) // set image to display`
  - `.setOnClickListener { … }`

Customize TextEdit with inputType

- **textPersonName**: Single line of text
- **textCapCharacters**: Set to all capital letters
- **textPassword**: Conceal an entered password
- **number**: Restrict text entry to numbers
- **textEmailAddress**: Show keyboard with @ conveniently located
- **phone**: Show a numeric phone keypad



ID: name_text

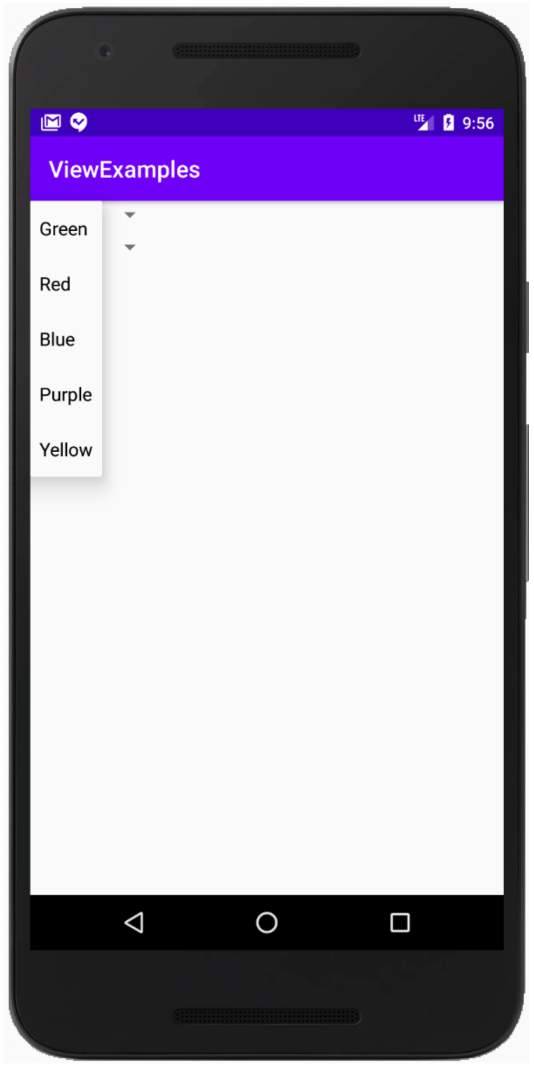
inputType

- ☐ none
- ☐ text
- ☐ textCapCharacters
- ☐ textCapWords
- ☐ textCapSentences
- ☐ textAutoCorrect
- ☐ textAutoComplete
- ☐ textMultiLine
- ☐ textImeMultiLine
- ☐ textNoSuggestions
- ☐ textUri
- ☐ textEmailAddress
- ☐ textEmailSubject
- ☐ textShortMessage
- ☐ textLongMessage
- ☑ textPersonName
- ☐ textPostalAddress
- ☐ textPassword
- ☐ textVisiblePassword
- ☐ textWebEditText
- ☐ textFilter
- ☐ textPhonetic
- ☐ textWebEmailAddress
- ☐ textWebPassword
- ☐ number
- ☐ numberSigned
- ☐ numberDecimal
- ☐ numberPassword
- ☐ phone
- ☐ datetime
- ☐ date
- ☐ time

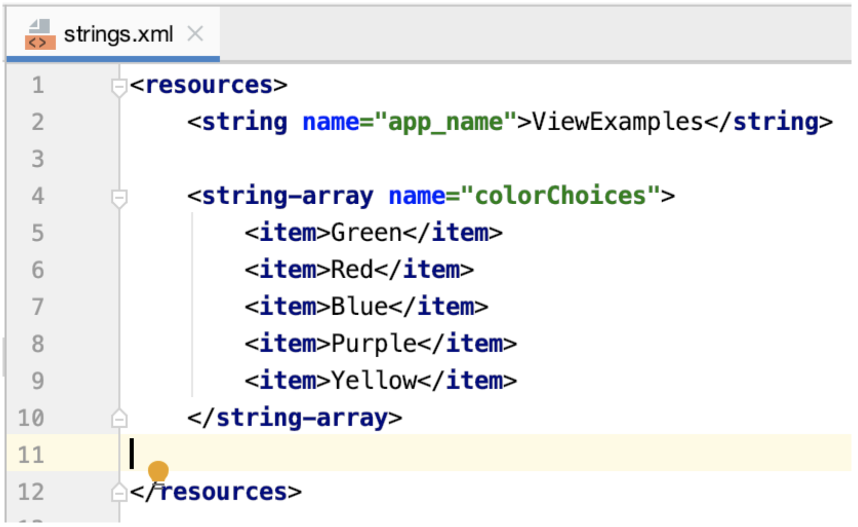
Views (Attributes and Listeners)

- **Switch (on/off)**
  - `.checked = booleanVal – set check state`
  - `.setOnCheckedChangeListener { … }`
- **Spinner (dropdown list)**
  - `.setAdapter(ArrayAdapter) – specify list values`
  - `.setSelection(int) – specify selected item`
  - `onItemSelectedListener { … }`
- **SearchView**
  - `queryHint –text to display when the field is empty`
  - `.setOnQueryTextListener { … }`

Setting Entries of a Spinner in the XML Layout File



```
<Spinner
    android:id="@+id/colorSelector1"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_marginEnd="32dp"
    android:layout_marginBottom="4dp"
    android:entries="@array/colorChoices"/>
```

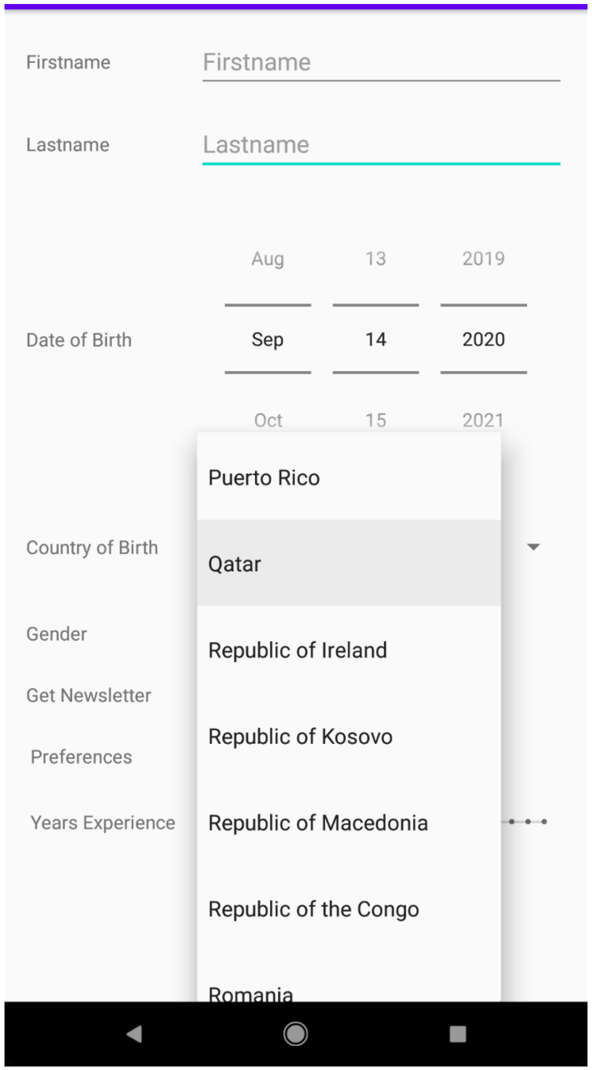


```
<resources>
    <string name="app_name">ViewExamples</string>

    <string-array name="colorChoices">
        <item>Green</item>
        <item>Red</item>
        <item>Blue</item>
        <item>Purple</item>
        <item>Yellow</item>
    </string-array>

</resources>
```

Setting Entries of a Spinner in Code



Firstname Firstname

Lastname Lastname

Date of Birth

Aug	13	2019
Sep	14	2020
Oct	15	2021

Country of Birth

Gender

Get Newsletter

Preferences

Years Experience

Puerto Rico
Qatar
Republic of Ireland
Republic of Kosovo
Republic of Macedonia
Republic of the Congo
Romania

```
<Spinner
    android:id="@+id/countriesSp"
    android:layout_width="0dp"
    android:layout_height="wrap_content"
/>
```

```
override fun onCreate(savedInstanceState: Bundle?) {
    super.onCreate(savedInstanceState)
    setContentView(R.layout.activity_register)

    CountryRepository.loadCountries(this)

    val adapter = ArrayAdapter<String>(
        this,
        android.R.layout.simple_dropdown_item_1line,
        CountryRepository.countryNames
    )
    countriesSp.adapter = adapter
}
```

Which View gets focus next?

- Topmost view on the activity layout
- After user submits input, focus moves to nearest neighbor—priority is left to right, top to bottom
- Arrange input controls in a layout from left to right and top to bottom in the order you want focus assigned
- Specify ordering in XML

```
    android:id="@+id/top"
    android:focusable="true"
    android:nextFocusDown="@+id/bottom"
```

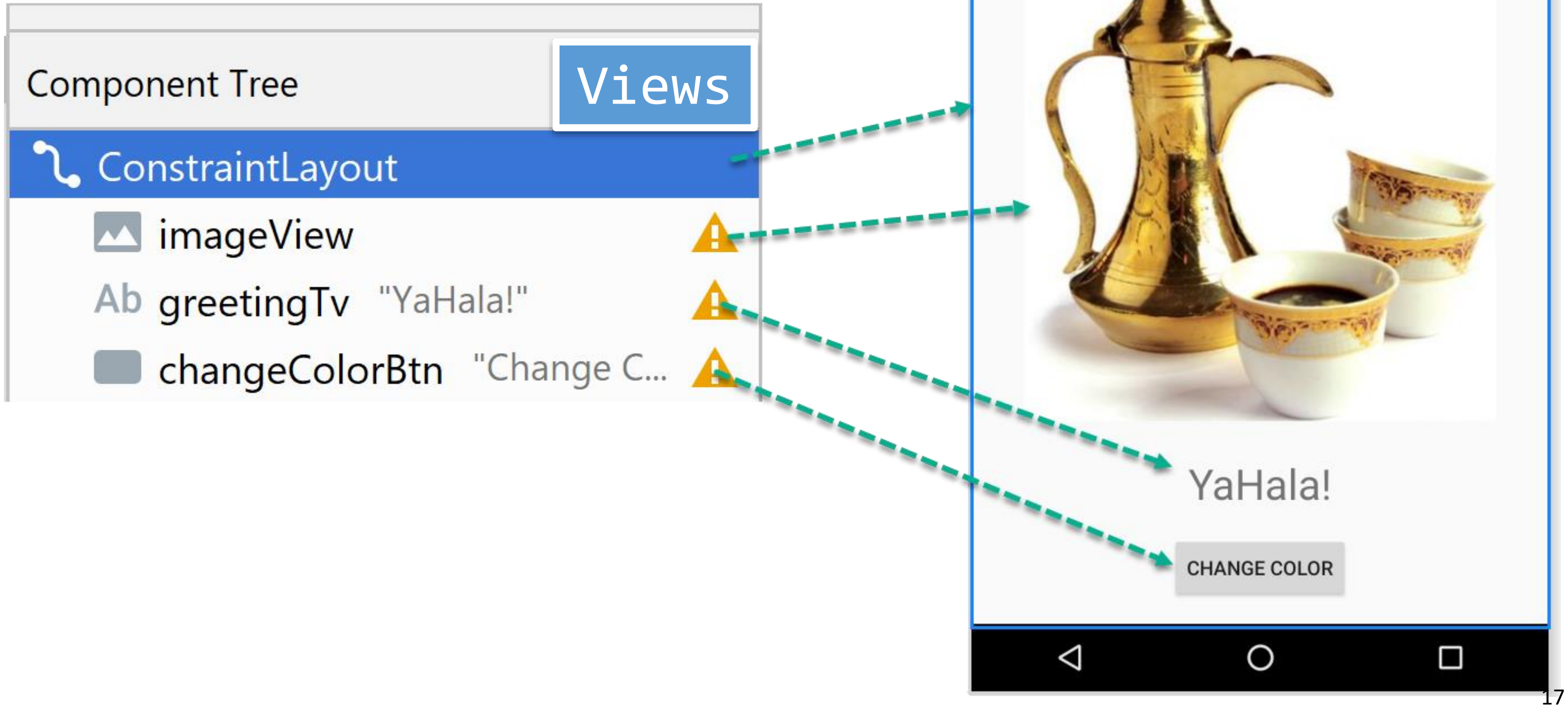
Set focus explicitly

Use methods of the View class to set focus

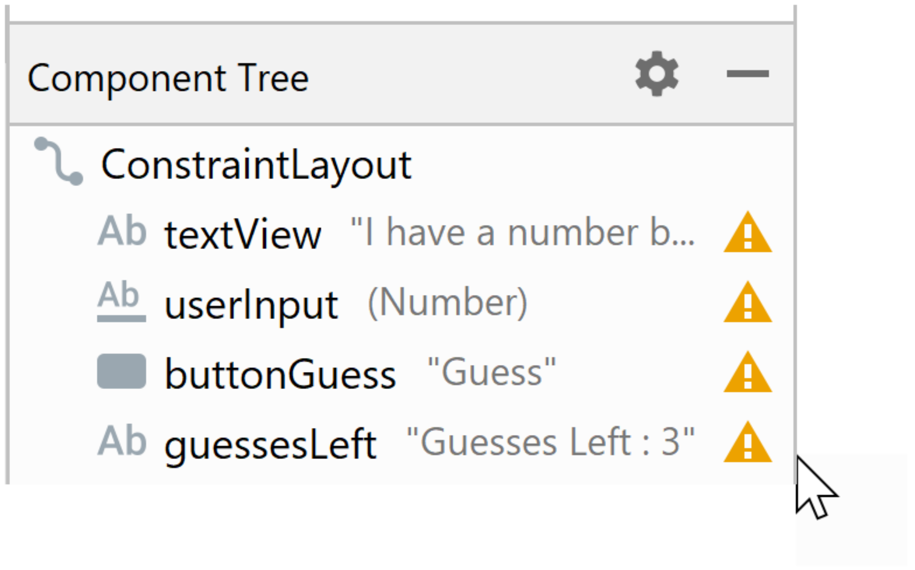
- setFocusable() sets whether a view can have focus
- requestFocus() gives focus to a specific view
- setOnFocusChangeListener() sets listener for when view gains or loses focus
- Find the view with focus
    - Activity.getCurrentFocus()
    - ViewGroup.getFocusedChild()

App 1 - Color Changer

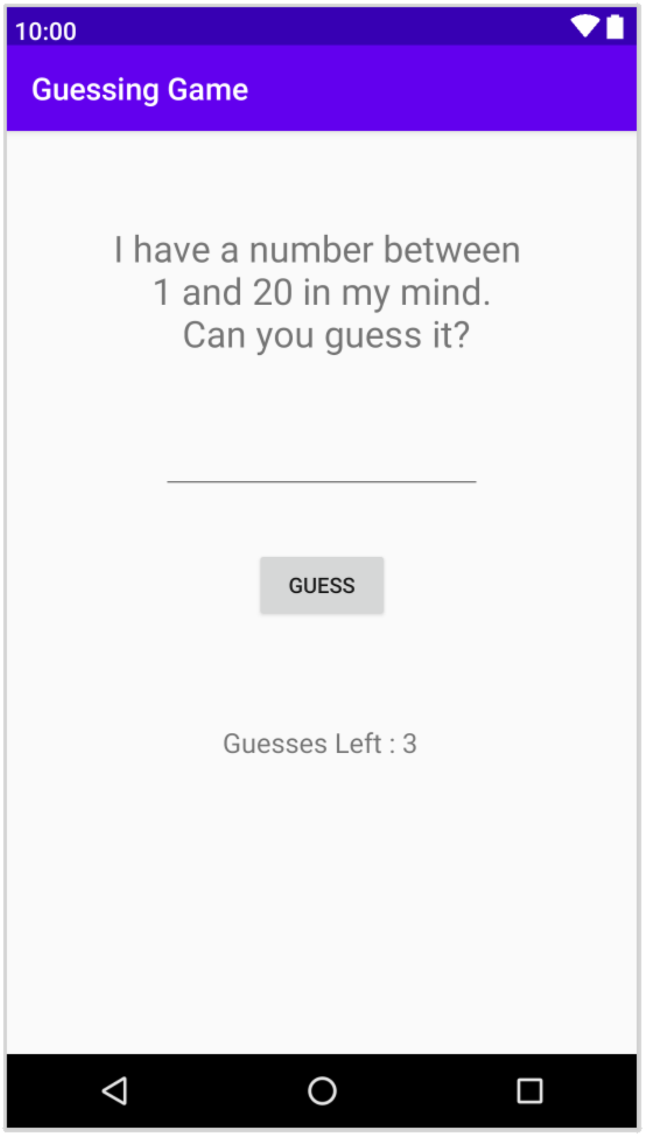
App that contains Text reading "YaHala!", an Image and a `Button` that randomly changes text's color with every click



App 2 – Guessing Game


Component Tree

ConstraintLayout
- textView "I have a number b...
- userInput (Number)
- buttonGuess "Guess"
- guessesLeft "Guesses Left : 3"


10:00

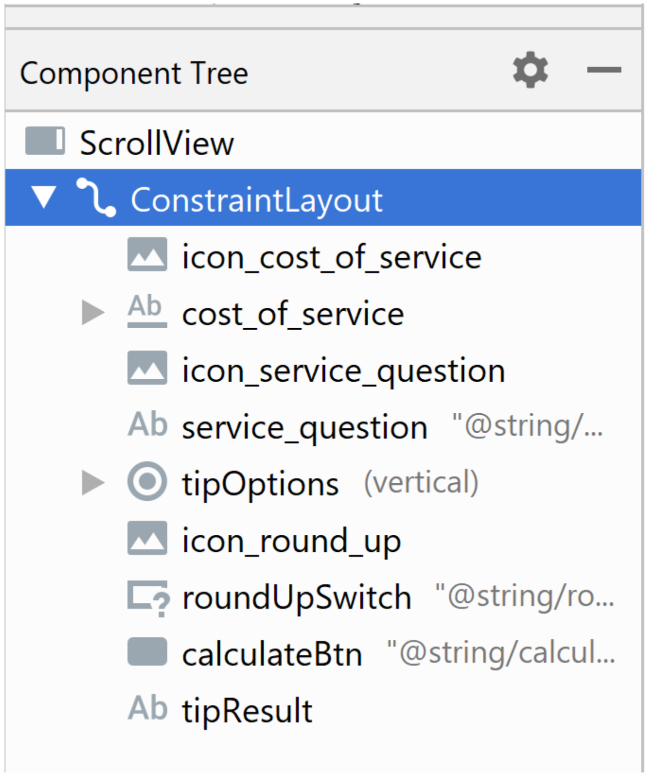
Guessing Game

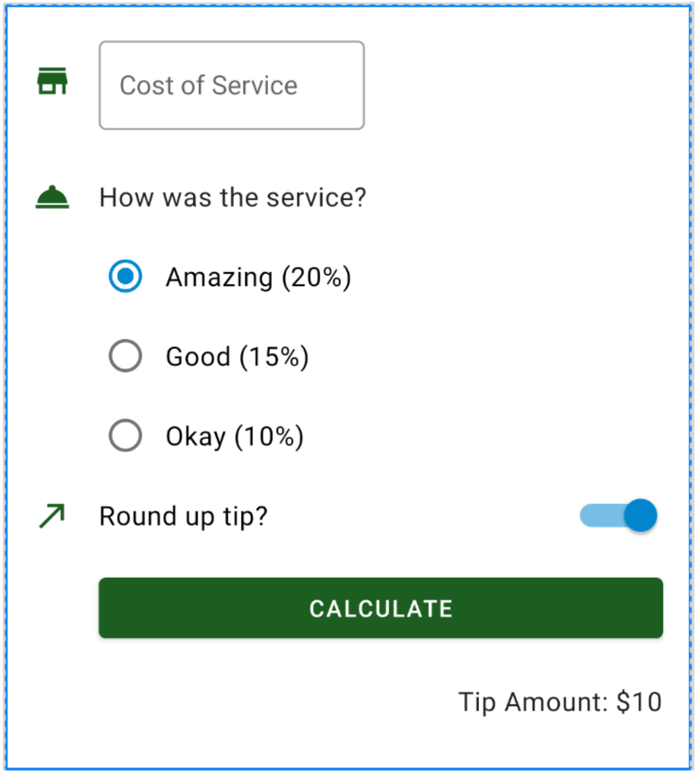
I have a number between
1 and 20 in my mind.
Can you guess it?

GUESS

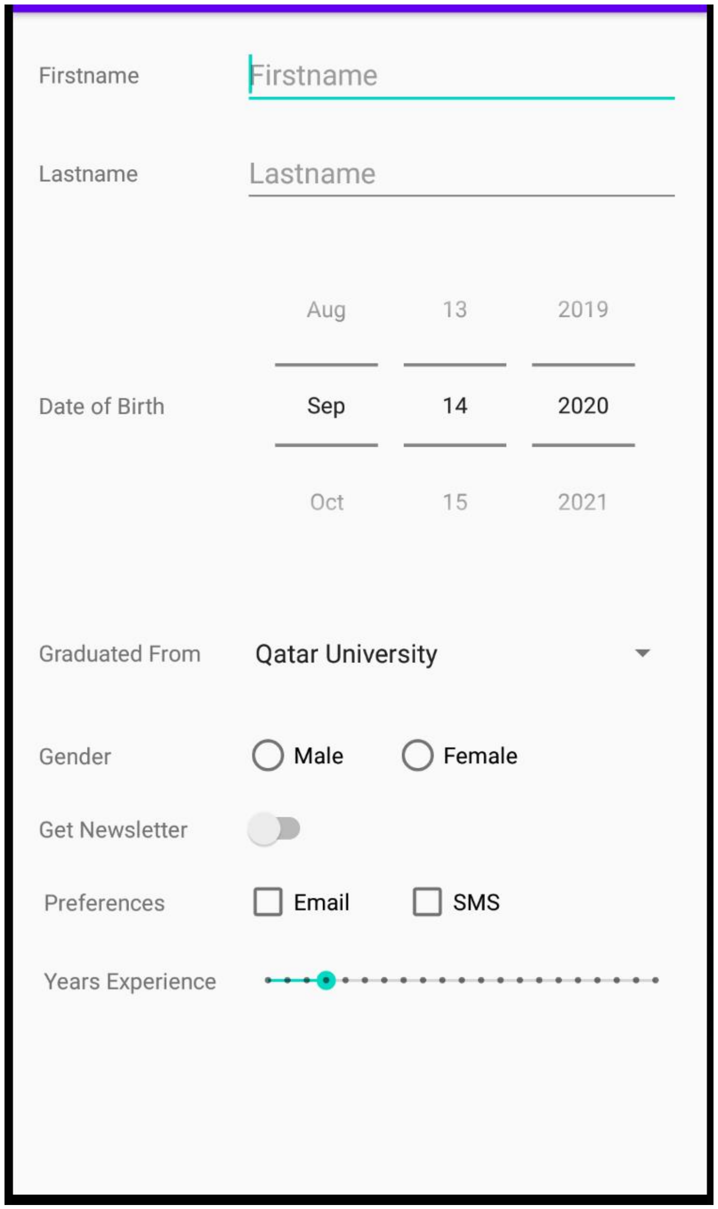
Guesses Left : 3

App 3 – Tips Calculator





Registration Form



Firstname	Firstname
Lastname	Lastname
Date of Birth	Aug 13 2019 / Sep 14 2020 / Oct 15 2021
Graduated From	Qatar University
Gender	○ Male ○ Female
Get Newsletter	
Preferences	☐ Email ☐ SMS
Years Experience	

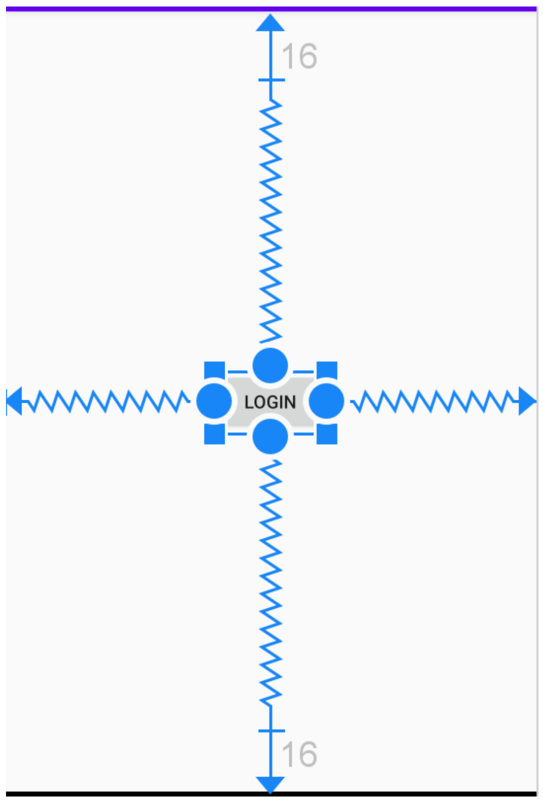
Material Design Components

- Using MDC to make your app look great easily

https://material.io/components

  - Float labels – TextInputLayout
  - FloatingActionButton
  - NavigationDrawer
  - Toolbar
  - CardView
  - TabLayout
  - BottomNavigationView
  - BottomSheet
  - Snackbar

Constraint Layout



Back

Responsive UI

- Layout automatically **controls** the **size** and **placement** of views to create a **Responsive UI**
    - Frees programmer from handling/hardcoding the sizing and positioning of UI elements
    - **Responsive UI =** When the screen is resized, the views reorganize themselves based on the rules of the layout

Constraint Layout

- ConstraintLayout: Allows buliding a Responsive UI by defining constraints for views
  - A constraint is a **connection** to another view, parent layout, or invisible Guideline / Barrier
  - Constraints control the **position** and **alignment** of UI elements

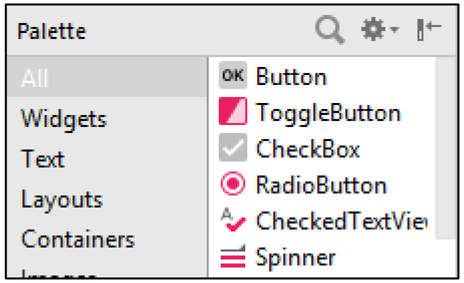
    - Position a view relative others including the parent

    - Need to add at least one horizontal and one vertical constraint

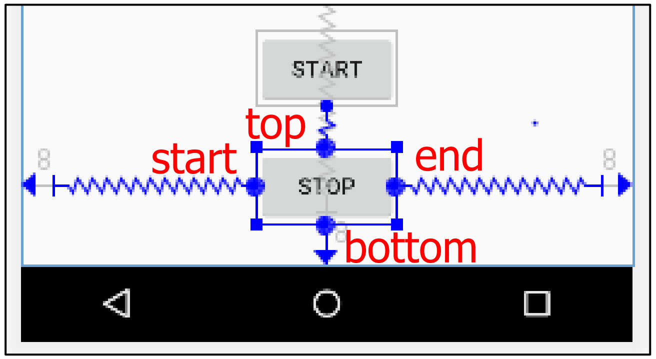
    - Center views by adding constraints to opposite sides of the view

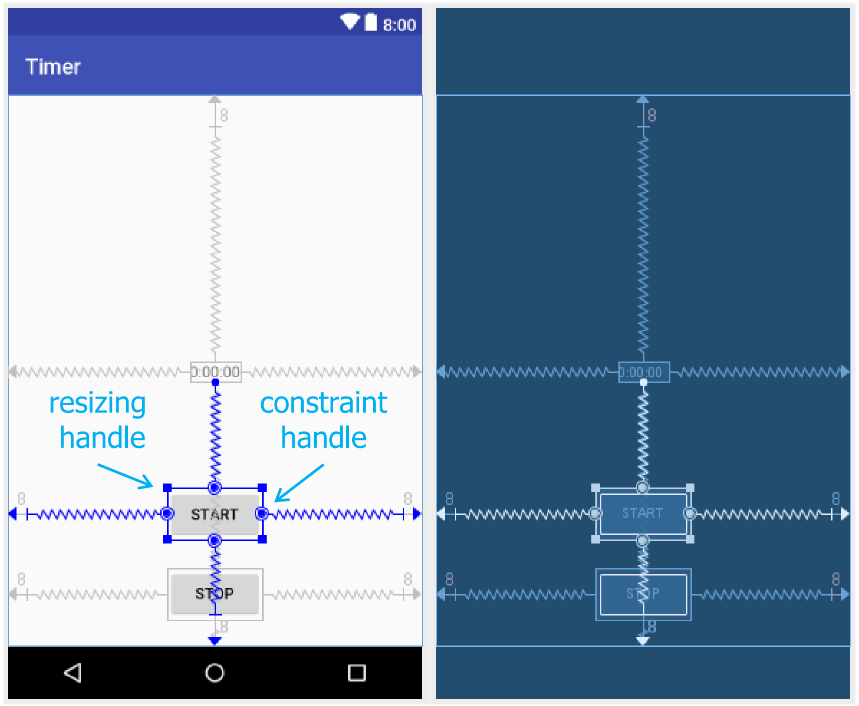
Defining Constraints

Steps
1. Drop a view to the editor
2. Connect constraint handles
   (e.g., top/bottom/left/right)



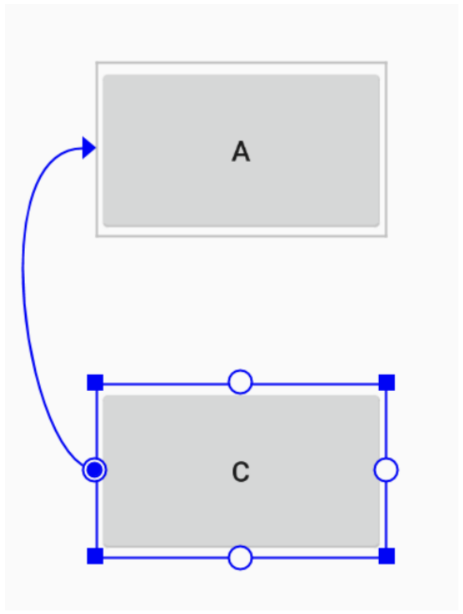
Add at least one horizontal and one vertical constraint





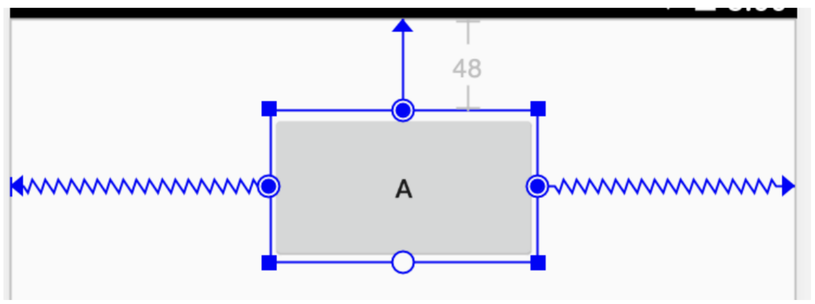
Alignment

- Align the edge of a view to the same edge of another view
- The left side of C is aligned to the left side of A.
  - If you want to **center** view C, create a constraint on both sides

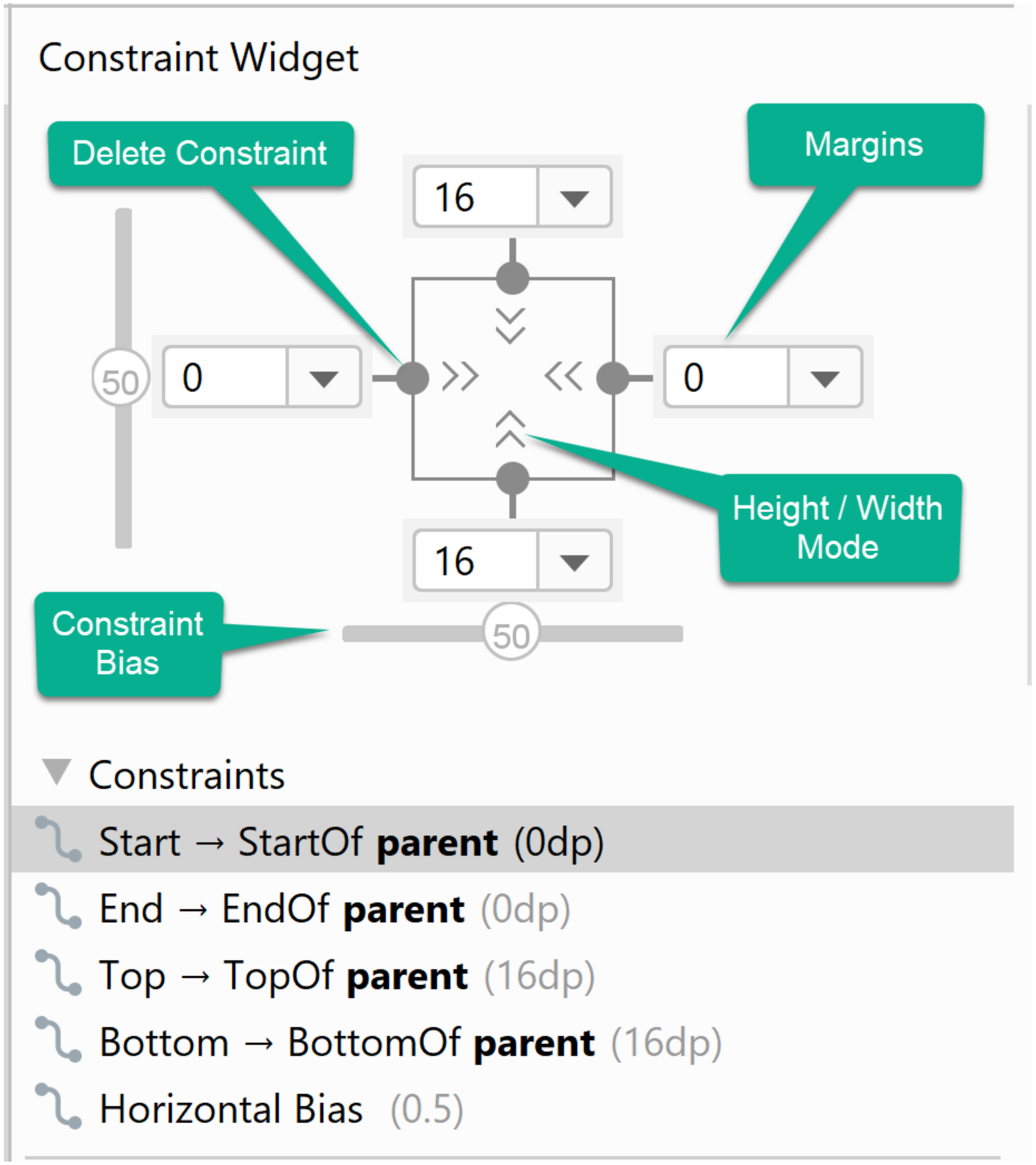


Bias

- If you add opposing constraints on a view, the constraint lines become like a **spring** to indicate the opposing forces
- The view becomes centered between the two constraints with a bias of 50% by default
- You can adjust the bias by dragging the view



View Constraints Editor


Constraint Widget

Constraints

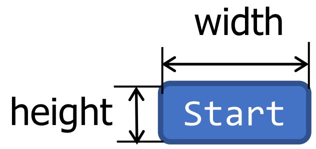
Start → StartOf **parent** (0dp)

End → EndOf **parent** (0dp)

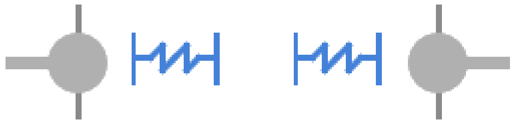
Top → TopOf **parent** (16dp)

Bottom → BottomOf **parent** (16dp)

Horizontal Bias (0.5)

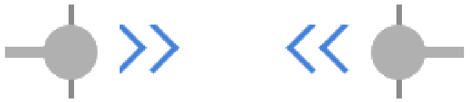


View Size



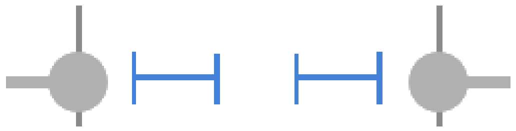
`layout_width="0dp"`

- The view expands to **match constraints** on each side (after accounting for the view's margins)
  - View will grow/shrink on resizing



`layout_width="wrap_content"`

- The view expands as needed to **fit** its contents

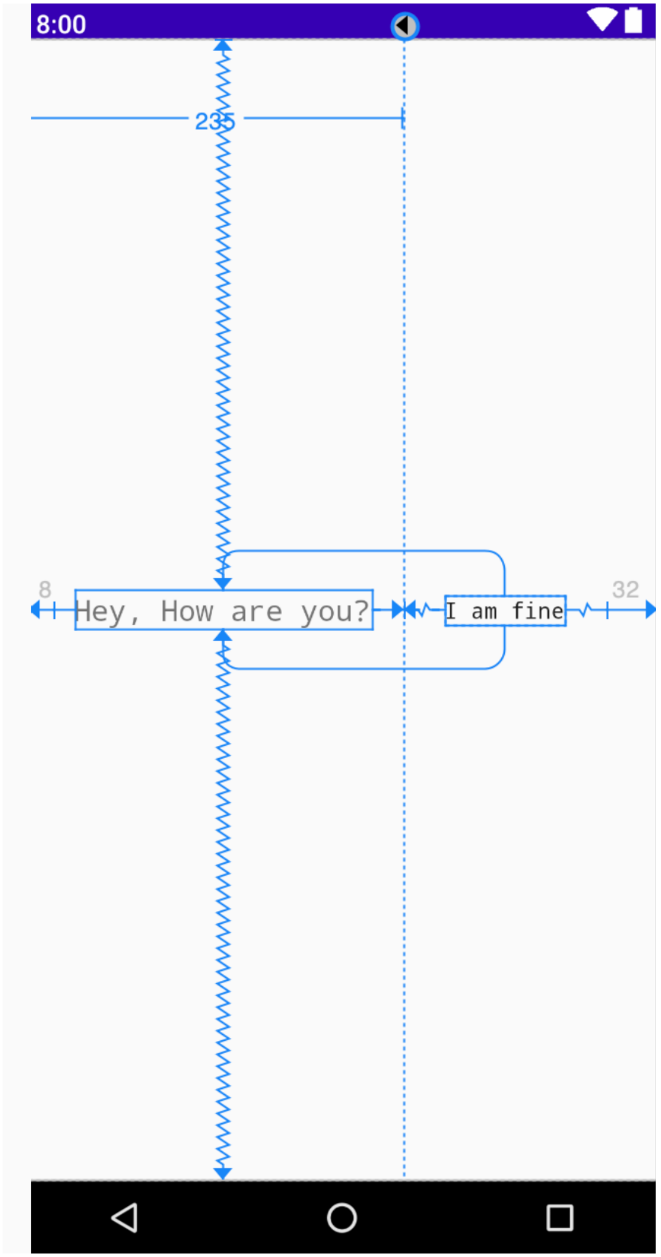


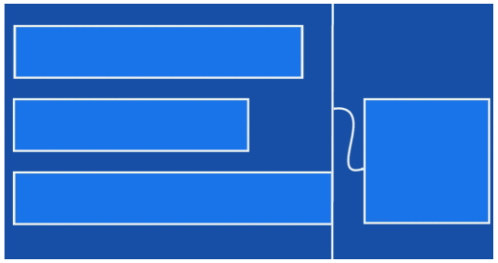
`layout_width="200dp"`

- **Fixed** size (e.g., 200dp density-independent pixels)

Guideline

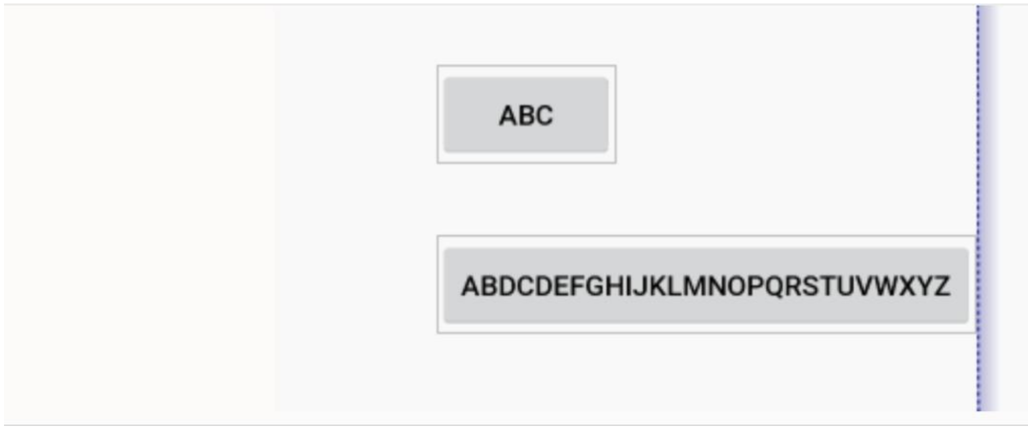
- A guideline is a visual guide used to divide the layout
- Add a vertical or horizontal **guideline** to which you can constrain views, and the guideline will be invisible to app users
- Position the guideline within the layout based on either **dp** (Density-independent pixels) units or percent relative to the layout's edge

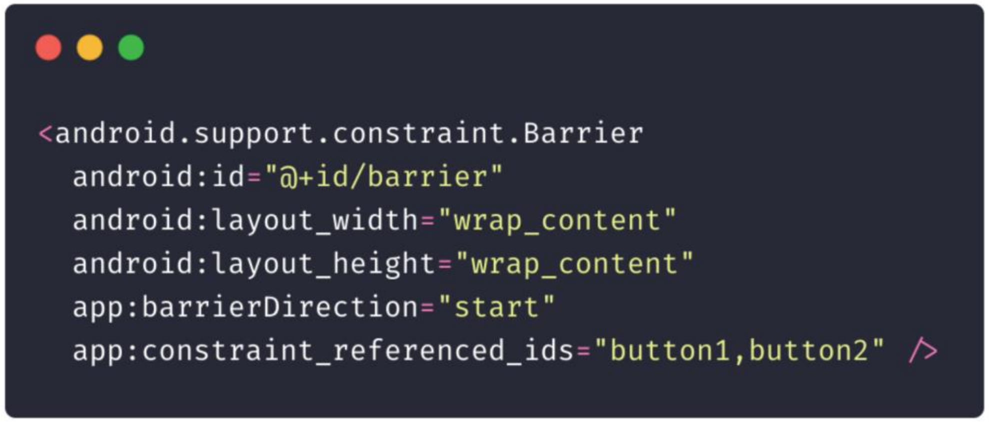




Barrier

- A Barrier is a virtual view, similar to a Guideline, to which we can constrain objects
- The Barrier width/height are determined by the views placed in it
- Use a barrier any time the views placed in it **could dynamically vary in size** based on user input or language setting

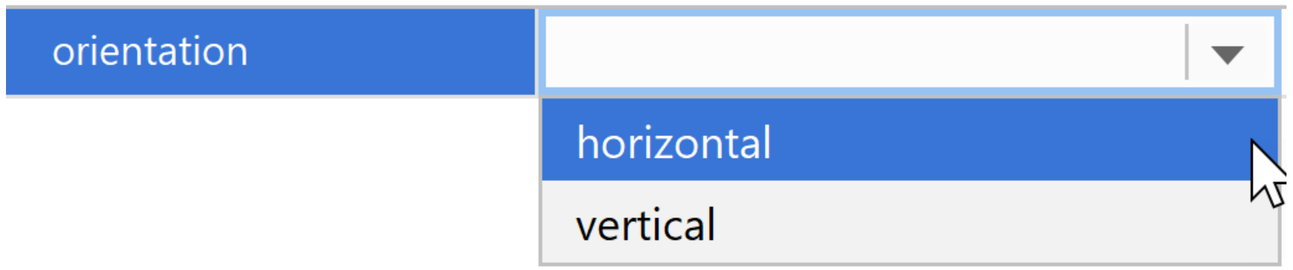


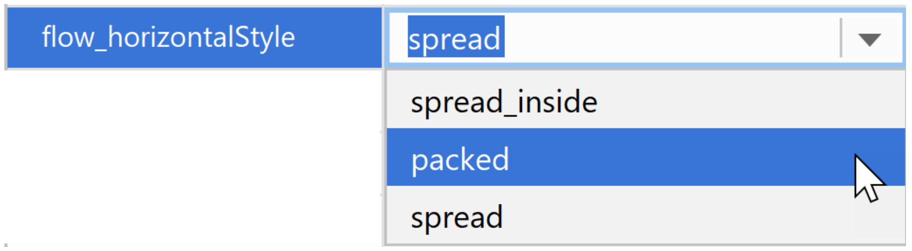


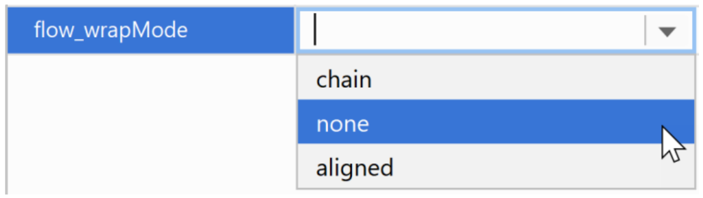
```
<android.support.constraint.Barrier
  android:id="@+id/barrier"
  android:layout_width="wrap_content"
  android:layout_height="wrap_content"
  app:barrierDirection="start"
  app:constraint_referenced_ids="button1,button2" />
```

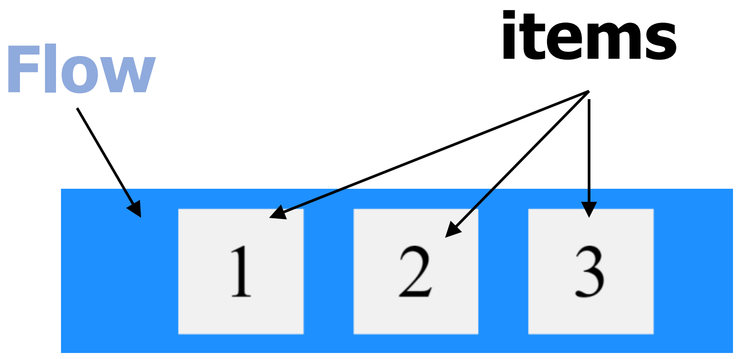
Flow

- Flow provides an efficient way to **distribute space** among items in the flow while accommodating different screen sizes



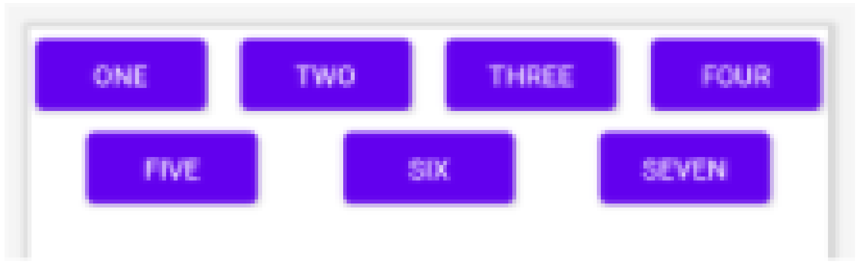




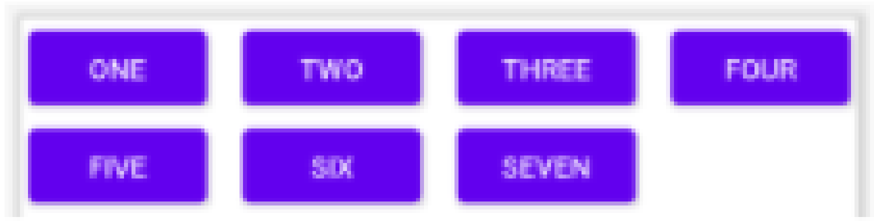


Flow - wrapMode

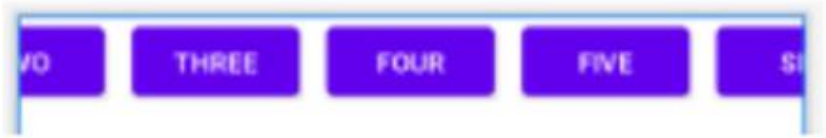
```
app:flow_wrapMode = "none | chain | aligned"
```



```
app:flow_wrapMode="chain"
```



```
app:flow_wrapMode="aligned"
```



```
app:flow_wrapMode="none"
```

Reusing Layouts

- Extract commonly used elements into common layout and then use `<include>` tag to include a layout

```
<include
    android:id="@+id/toolbar"
    layout="@layout/toolbar"
    android:layout_width="match_parent"
    android:layout_height="wrap_content" />
```

Summary

- **Activity** provides the UI that the user interacts with
  - It has layout (.xml) file & Activity class (UI Controller) => This allows a **clear separation** between the UI and the app logic
  - Activity class define listeners to handle events
- ConstraintLayout enables responsive design

.. mastering it will take some time and practice 🧗 🏋 ...

Resources

- Build a Responsive UI with ConstraintLayout
    - https://developer.android.com/training/constraint-layout
- ConstraintLayout codelab
    - https://codelabs.developers.google.com/codelabs/constraint-layout/
    - https://developer.android.com/codelabs/kotlin-android-training-constraint-layout
- Android Dev Guide
    - https://developer.android.com/guide/